AMENDMENTS TO THE SPECIFICATION

Please replace the paragraph starting at page 3, line 29, and ending at page 3, line 30, with the following amended paragraph:

FIG. 8 illustrates an embodiment of the present invention having multiple packet producers to support a variety of data formats;

Please replace the paragraph starting at page 4, line 17, and ending at page 5, line 7, with the following amended paragraph:

Directing attention to FIG. 1, the present invention utilizes a client-server computer architecture 100 implemented over a large, public network such as the Internet. Server 102 is responsible for distributing streaming media assets such as video, audio, static images, graphics, or a combination thereof to clients 104-1, 104-2,...,104-n, where n is the number of clients requiring streaming media assets, via public computer network 106. Media assets are streamed by transmitting a sequence of packets from the server 102 to the client 104, which plays the media asset on a computer monitor, or over an audio device, or other suitable device using a media player program that decompresses, decodes, and performs any necessary processing on the sequence of packets received from the server 102 to present aural or visual presentation contained in the packets to a user. Examples of streaming media assets include movies, newscasts, music, graphics, animation, slide presentations, and the like, all of which are capable of being presented in a serial fashion to a human user. Server 102 may be a source of the streaming media assets. Optionally, one or more third party content providers such as content provider 108 may be in communication with server 102, and provide the streaming media assets to the server 102 over network 106. Media assets are typically stored in files in the memory of the server 102 and distributed to clients on demand or according to a schedule. A property that is common to various techniques for streaming of variable bit rate multimedia data involves delivering variable size data packets at discrete points in time (with variable interval) over a stream's delivery timeline. As an example, consider FIG. 5 which shows data packets P1, P2, etc. that are due at times t1, t2, etc. The packets are of different sizes and the time interval between t1, t2, etc. varies. Contrast this with constant bit rate (CBR) delivery which involves streaming fixed size packets of data over fixed time intervals.

Please replace the paragraph starting at page 5, line 8, and ending at page 5, line 25, with the following amended paragraph:

FIG. 2 illustrates in block diagram form the major components included in a computer embodying either server 102 or client 104. Computer 150 incorporates a processor 152 such as a central processing unit (CPU) and supporting integrated circuitry. In an embodiment, work stations such as Sun Ultra computers available from Sun Microsystems, Inc., of Santa Clara, CA, can be used as server 102. Personal computers such as those available from Dell ~~Corporation~~ Inc., of Austin, TX, may be used for client computers 104. However, in general any type of computer may be used for a server 102 and any type of computer or even various information appliances may be used for ~~the~~ client 104. Memory 154 may include one or more of RAM and NVRAM such as flash memory, to facilitate storage of software modules executed by processor 152, and file systems administering media assets. As referred to herein, a file system refers to any administrative entity implemented by computer 150 to organize and administer media assets. File systems can include conventional file systems, direct attached storage, network attached storage, storage area networks, both block based and file based, raw storage, and the like. Also included in computer 150 are keyboard 156 or other input device, pointing device 158, and monitor 160, which allow a user to interact with computer 150 during execution of software programs. Mass storage devices such as disk drive 162 and CD ROM 164 may also be in computer 150 to provide storage for computer programs, associated files, and media assets. In one embodiment, database products available from Oracle Corp. of Redwood Shores, CA, may be utilized in connection with file systems as a database and database server.

Please replace the paragraph starting at page 5, line 25, and ending at page 6, line 3, with the following amended paragraph:

Computer 150 communicates with other computers via communication connection 166 and communication line 168 to allow ~~the~~ computer 150 to be operated remotely, or utilize files stored at different locations, such as content provider 108. Communication connection ~~156~~ 166 can be a modem, network interface card, or other device that enables a computer to communicate with other computers. Communication line 168 can be a telephone line or cable, or any medium or channel capable of transferring data between computers. In alternative embodiments, communication connection 166 can be a wireless communication medium, thus eliminating the need for

communication line 168. The components described above may be operatively connected by a communications bus 170.

Please replace the paragraph starting at page 6, line 3, and ending at page 6, line 19, with the following amended paragraph:

FIG. 3 illustrates functional components included in an embodiment of server 102. Server 102 includes delivery system 200, which obtains content in the form of packetized streaming media assets and delivers it to ~~the~~ clients 104. Non-CBR streaming media assets are supported by ~~the~~ delivery system 200 as well as streaming media assets having fixed packet sizes and delivery times. As used herein, non-CBR streaming refers to the delivery of variable size packets of data at variable time intervals. That is, packet sizes can vary, and the time interval between packets could vary as well. Delivery system 200 supports non-CBR streaming media assets by associating a time stamp with each packet, and delivering the packet either at or before the time specified in the time stamp. By using time stamped packets, delivery of the packets can be made without missing any deadlines. As illustrated in the FIG. 4, such on time delivery is shown where packet P1 is delivered at time t1, packet P2 is delivered at time t2 and so on. In addition to delivering packets of uniform size at regular intervals, ~~the~~ delivery system 200 also supports delivery of variable sized packets at fixed intervals in time (FIG. 5) such as when delivering I, P, or B frame data every 1/30th of a second. Delivery of fixed size packets at variable intervals in time (FIG. 6) is supported, such as needed by ASF slide presentation authoring tools that attempt to build nearly the same sized packets with very small variation.

Please replace the paragraph starting at page 6, line 20, and ending at page 6, line 31, with the following amended paragraph:

While the use of packets with time stamps allows ~~the~~ delivery system 200 to support non-CBR delivery, the ability to handle a variety of formats, inputs sources, etc. is desirable. ~~The d~~Delivery system 200 utilizes packet producer 202 that can service a variety of input sources such as data read from a file, data received from ~~the~~ network 106, data read from a circular disk buffer while synchronizing with another capture process, and the like. Packet producers 202 are implemented as software modules that acquire data to be streamed to ~~the~~ clients 104, parse the acquired data if necessary, and produce time stamped packets for delivery. ~~The p~~Packet producers 202 can be specialized to handle specific formats by including, for example, code that parses Quicktime files, locates the hint tracks and constructs the realtime transport

protocol (RTP) packets or code that parses ASF files and locates the index entries that are at the end of the file, etc. By providing a plurality of specialized packet producers 202 in ~~the~~ delivery system 200, the delivery system can handle data in any anticipated format.

Please replace the paragraph starting at page 7, line 1, and ending at page 7, line 9, with the following amended paragraph:

The time stamped packets produced by the packet producer 202 are sent from the packet producer 202 to the time stamped packet queue 204, a data structure that organizes time stamped packet<u>s</u> into a first in, first out queue. While the packet producer 202 is a producer of time stamped packets, a feeder software module 206 removes the packets from the queue and delivers them to the client according to the time stamp on each packet. In an embodiment, both packet producers <u>202</u> and feeders <u>206</u> are active entities (with an associated thread) and the time stamped packet queue 204 is a passive data structure. Each time stamped packet in the queue doesn't need to contain the packet data in the queue verbatim, but only a pointer to where the data is stored, such as in a buffer 208 that is shared with the packet producers 202.

Please replace the paragraph starting at page 7, line 10, and ending at page 7, line 16, with the following amended paragraph:

In an embodiment, a packet producer 202 includes two software components: a stream reader 210 and a stream processor 212. ~~The s~~<u>S</u>tream reader 210 produces the data stream by receiving data and sending it to ~~the~~ stream processor 212. ~~The s~~<u>S</u>tream processor 212 takes the data from ~~the~~ stream reader 210, parses it if necessary, and produces time stamped packets. Both ~~the~~ stream reader 210 and ~~the~~ stream processor 212 are software components that run in a common thread, with ~~the~~ stream processor 212 calling ~~the~~ stream reader 210 whenever it needs more data.

Please replace the paragraph starting at page 7, line 17, and ending at page 7, line 28, with the following amended paragraph:

Since the stream processors are format specific in an embodiment, they can be employed to modify the stream when needed. While only three packet producers are shown in FIG. 3, it is to be understood that any number of packet producers can be included and configured to accommodate whatever data and delivery requirements are

contemplated. Examples of packet producers include a packet producer for processing ASF files from disk or other storage devices, a packet producer for processing live ASF streams from the network 106, a packet producer for processing Quicktime files from disk, a packet producer for processing Quicktime files that have ready made RTP packets, a packet producer for processing ASF streams from shared memory, a packet producer for processing MPEG-4 from disk based circular capture buffer, and a packet producer for MPEG-4 that injects special trailers or modifies the stream in a special way. Stream processors can act as demultiplexers (separating audio and video from a single stream, for example) and feed multiple time stamp packet queues as shown in FIG. 7.

Please replace the paragraph starting at page 7, line 29, and ending at page 8, line 10, with the following amended paragraph:

The pPacket producers 202 are responsible for dealing with any indexing information that the data might have. The index information may be part of the file itself, for example, such as with ASF, or it could be in a separate file, such as implemented in the current MediaBase system available from Kasenna, Inc., of Mountain View, CA. The packet producers can also hide how fast-forward/rewind is implemented and provide flexibility in different ways of supporting fast-forward/rewind. For example, the packet producer can decide whether to use a separate file for supporting FF/REW, or generate the FF/REW stream from the main file on-the-fly. It should be noted that the time stamp added to a packet by a packet producer is meant to be used by the feeder only, and doesn't necessarily correspond to the exact time of presentation as designated in the original media stream. In the case of reverse-play (rewind), the time stamps seen on the packets in the original media stream are decreasing when the stream is traversed in the reverse order. However, the time stamps produced by the packet producers for this reverse stream will always be increasing as these time stamps correspond to the delivery time to be used by the feeder.

Please replace the paragraph starting at page 8, line 11, and ending at page 8, line 26, with the following amended paragraph:

In an embodiment, there is one packet queue per active stream in the delivery system 200. One packet producer is placing packets into a packet queue at a given time, and one feeder is removing packets from the packet queue at a given time. Typically, a feeder will deliver multiple streams, and hence will deliver packets from

multiple packet queues. As shown in FIG. 8, packet producer 220 produces QuickTime packets and places them into time stamp packet 222, packet producer 224 produces ASF packets and places them into time stamp packet queue 226, packet producer 228 produces MPEG-4 packets and places them into time stamp packet queue 230, and packet producer 232 produces QuickTime packets and places them into time stamp packet queue 234. Time stamp packet queues 222, 226, 230, and 234 are serviced by feeder 240, which disburses packets from time stamp packet queue 222 to client 242, packets from time stamp packet queue 226 to client 244, packets from time stamp packet queue 230 to client 246, and packets from time stamp packet queue 234 to client 248. However, as shown in FIG. 3, a delivery system 200 can have more than one feeder, such as when one feeder cannot take up additional work, and an additional processor is available on which an additional feeder can be started. In such a case, another feeder for processing the extra load is instantiated. Having multiple feeders may not be necessary on a machine having a single processor.

Please replace the paragraph starting at page 10, line 15, and ending at page 10, line 25, with the following amended paragraph:

Traditionally, client side buffers located in memory 154 of ~~the~~ client 104 are used to smooth out the jitter in the arrival rate of data at the client side. There are two parameters that are critical in client side buffering. They are (i) the amount of data pre-read before the playout starts (pre-read size), and (ii) the size of the client side buffer (max buffer size). The pre-read size and max buffer size parameters impose the maximum limits on how late or how early a packet can arrive. When media is streamed at a fairly constant rate, if the arrival rate of data into the buffer matches the consumption rate of data by the decoder, then there should be pre-read size data left in the buffer. However, since the data can arrive late or early, buffering helps. The pre-read size data in the client's buffer protects against buffer underflow, if the data is not received in time. The max buffer size protects against overflow of the client's buffer if data starts arriving earlier than expected.

Please replace the paragraph starting at page 10, line 26, and ending at page 11, line 2, with the following amended paragraph:

Embodiments of the present invention allow a packet's time stamp to be adjusted based on the client side pre-read size and/or max buffer size parameters. Directing attention to FIG. 14, when a client 104 requests delivery of a media asset

from the server 102, the server 102 can query the client 104 for values corresponding to the client's pre-read size and max buffer size parameters. If it is known that a client pre-reads one second's worth of data, then a packet going to that client can be delayed up to a maximum of one second. Also, if it is known that a client 104 has a max buffer size to hold up to, for example, 10 seconds worth of data, then a packet can be sent to the client 104 as early as 10 seconds ahead of its timestamp.

Please replace the paragraph starting at page 11, line 3, and ending at page 11, line 18, with the following amended paragraph:

FIGS. 14A and 14B illustrate the sequences of steps executed by embodiments of the present invention to perform conflict resolution between media asset streams having conflicting time stamps. At step 350, the streaming media assets are admitted by the server 102 for transmission to clients 104. The delivery times of the packets in the individual media asset streams are compared in step 352. If no conflict is discovered between the time stamps of the streaming media assets admitted for delivery (decision step 354), control proceeds to step 364 and the packets are transmitted to the clients 104. If a conflict is discovered at decision step 354 such that two or more packets have matching time stamps, control proceeds to step 356 (FIG. 14A), where the client's pre-read size value is read by the server 102. The pre-read size value can be stored on the server 102 when a client makes a request for delivery of a media asset. In another embodiment (FIG. 14B), the max buffer size value can be checked (decision step 358) by the server in a similar manner. If the size of the value is sufficient to accommodate a delayed or earlier delivery of the packet (decision step 360), one of the conflicting time stamps is adjusted to a different, non-conflicting delivery time at step 362. Control proceeds to step 364, where the packets are transmitted to the clients 104. If there are remaining packets in the streams (decision step 366), control returns to step 352.

Please replace the paragraph starting at page 12, line 1, and ending at page 12, line 7, with the following amended paragraph:

Also, one second's worth of data for 800 Kbps stream is different from one second's worth of data for 1.5 Mbps stream. That is, the parameters pre-read size and max buffer size are related to the bit rate of the media asset. For example, the pre-read size and max buffer size parameter value can vary between movies that are streamed over the network 106. A request for delivery of a media asset, such as an openMovie

call made from the client to the server returns these parameters from the server to the client indicating how much buffer should be allocated by the client, and how much data should be pre-read before the playout starts.

Please replace the paragraph starting at page 14, line 1, and ending at page 14, line 4, with the following amended paragraph:

The timeWindowToBytes function scans the file and returns the maximum number of bytes that need to be delivered during a time window period. The spaceWindowToSeconds function scans the file and returns the shortest duration of a space window amount of bytes in the stream.